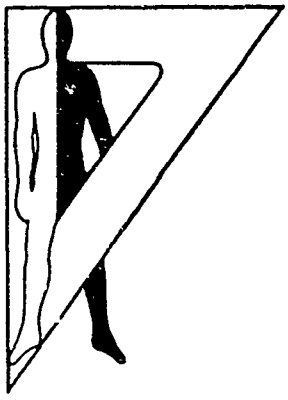


AD

Technical Memorandum 15-89

SOUND EFFECTS GENERATOR FOR USE WITH THE COCKPIT RESEARCH AND EXPERIMENTATION WORK LOAD SIMULATOR (CREWS)

Michael W. Thompson

October 1989
AMCMS Code 612716.H7000

Approved for public release;
distribution is unlimited.

U. S. ARMY HUMAN ENGINEERING LABORATORY
Aberdeen Proving Ground, Maryland

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Memorandum 15-89	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Human Engineering Laboratory 6b. OFFICE SYMBOL (If applicable) SLCHE	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Aberdeen Proving Ground, Maryland 21005-5001	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 6.2 / PROJECT NO. 1L162716AH70 / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Sound Effects Generator for Use With the Cockpit Research and Experimentation Work Load Simulator (Crews)

12. PERSONAL AUTHOR(S)
Thompson, Michael W.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	1989, October	54

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
01	03	03.12
14	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
sound effects generator; noise generation; digital sound generation; tone generation; Real Time (EDC); synthesized sound; noise (see reverse side)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This manual describes the sound effects generator (SEG) developed for Digital Equipment Corporation UNIBUS computers. It provides the operational and programming information for using the SEG on DEC VAX UNIBUS computers. The SEG can produce a variety of complex sounds through two independent channels. Each channel is capable of producing noise and three tones with mechanisms provided for mixing the noise and tones, controlling the amplitude of each tone, and shaping the envelope of each channel for final output. Being a register-oriented device, the SEG is ideal for real time applications, requiring very little computer processing time to generate and alter sounds. The SEG is built on a single quad-height UNIBUS Foundation Module and plugs directly into a small peripherals controller slot on the UNIBUS back plane. Application programming is accomplished using FORTRAN callable subroutines which initialize and alter the desired sounds. Keywords:

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Technical Reports Office	22b TELEPHONE (Include Area Code) (301)278-4478 / 22c OFFICE SYMBOL SLCHE-SS-IR

DD Form 1473, JUN 86 — Previous editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

AMCMS Code 612716.H7000 Technical Memorandum 15-89

SOUND EFFECTS GENERATOR FOR USE WITH THE COCKPIT RESEARCH AND EXPERIMENTATION WORK LOAD SIMULATOR (CREWS)

Michael W. Thompson

October 1989

APPROVED: John D. Weisz
JOHN D. WEISZ
Director
Human Engineering Laboratory

Approved for public release;
distribution is unlimited.

U.S. ARMY HUMAN ENGINEERING LABORATORY
Aberdeen Proving Ground, Maryland 21005-5001

CONTENTS

SOUND EFFECTS GENERATOR FOR USE WITH THE COCKPIT RESEARCH AND EXPERIMENTATION WORK LOAD SIMULATOR (CREWS)

INTRODUCTION

A two-channel sound effects generator (SEG) has been designed and developed by the U.S. Army Human Engineering Laboratory (USAHEL) for use with the Human Factors Research Simulator (HFRS). Although not designed to reproduce or model any particular sound, each channel on the SEG is capable of digitally synthesizing a wide variety of complex sounds under software control through individual noise and tone components. The SEG provides the mechanisms for independently creating and altering the amplitude and frequency for three separate noise and tone components for each channel. The SEG allows the noise and tones to be blended (mixed) and shaped, producing a single complex sound with multiple elements of noise and tones.

This manual provides a general description of the sound effects generator and of the software developed for its use. This manual is divided into two sections. The first section provides the information necessary in understanding the SEG hardware concepts for UNIBUS interfacing and sound generation. The second section provides the information necessary for using the SEG software.

To understand the software presented, the reader should be familiar with the FORTRAN-77 programming language and the Digital Equipment Corporation (DEC) VAX/VMS Operating System (OS). To fully comprehend all the material in this manual, the reader should also be familiar with digital electronic theory, electronic computer concepts, UNIBUS concepts, the M1710 UNIBUS Foundation Module, the VAX architecture, and the MACRO-11 programming language.

Additional information pertaining to the DEC UNIBUS and the AY-3-8912 Programmable Sound Generator can be obtained from the DEC Computer Interfacing Accessories and Logic Handbook, the DEC VAX Hardware Handbook, the DEC PDP11BUS Handbook, and the General Instruments Microelectronics Data Catalog.

GENERAL DESCRIPTION

The SEG consists of two programmable sound generators, each of which can be independently and dynamically altered under program control. Each PSG is capable of producing three separate channels of noise and tones. Each channel contains a mixer for blending the noise and tones as desired. Furthermore, each channel's output can be independently shaped (modulated) based on one of 10 distinct wave forms (see Appendix A). These wave forms control the amplitude, period, or frequency, creating an envelope of noise or sound and may be dynamically altered to produce a particular sound or audible effect. In addition, the three output channels are mixed to produce a single output of sound. In effect, three separate channels of noise or tones can be independently mixed, shaped, and controlled to produce a single complex sound.

Built on a single, quad-height module, the SEG plugs directly into a UNIBUS small peripheral controller (SPC) slot. Described here in two sections (see Figure 1), section one consists of the programmable sound generators that perform the actual sound generation. Section two consists of the UNIBUS controller logic and the 8-bit I/O port that allows the PSGs to communicate with the host or computer.

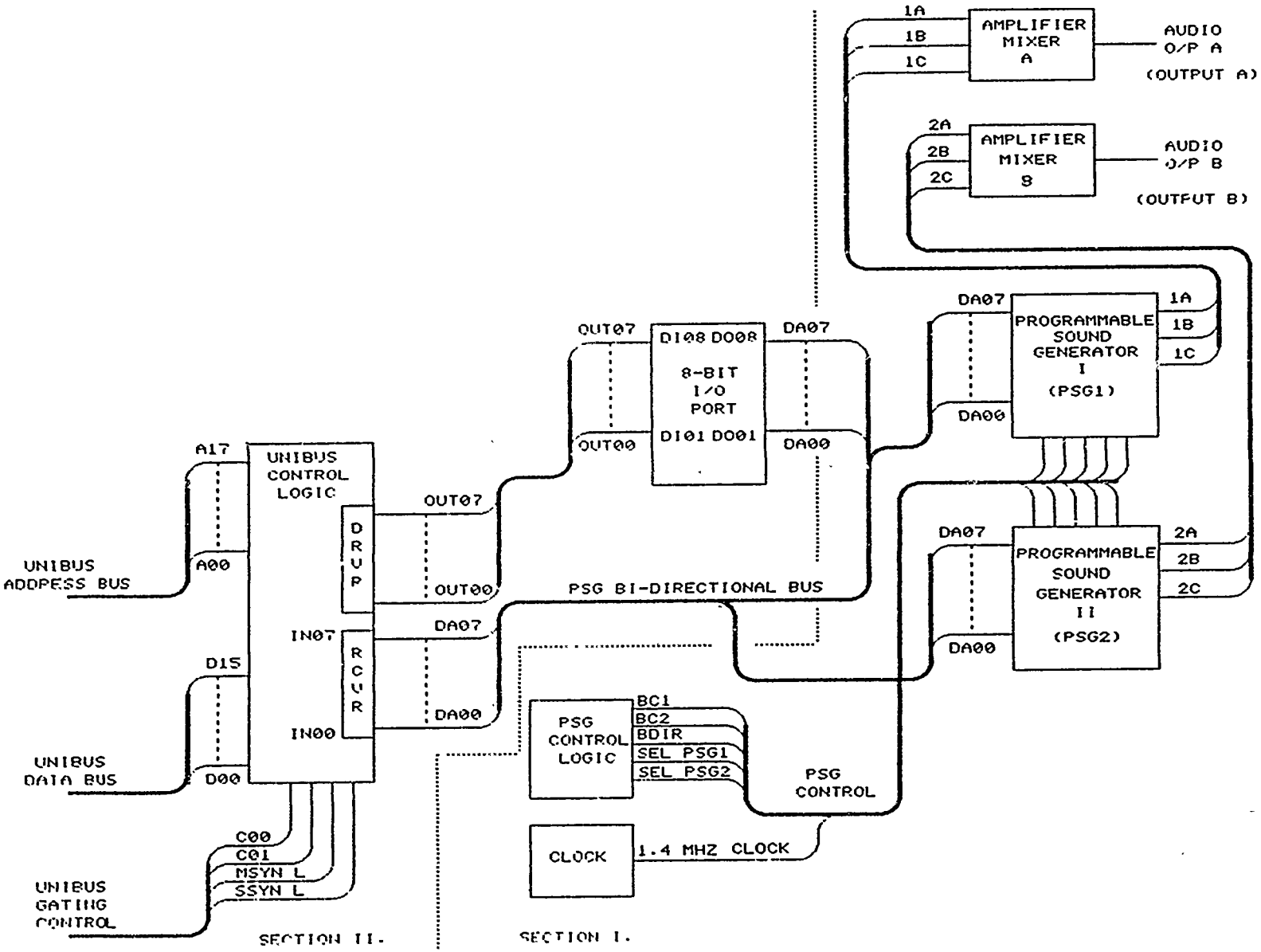


Figure 1. Sound effects generator block diagram.

For this application, the SEG is connected to a DEC VAX 11/750 computer using the VMS operating system. It should be noted, however, that the sound effects generator is not limited to use on the VAX 11/750 but may be installed on any computer system with the DEC UNIBUS architecture.

DETAILED DESCRIPTION

The SEG developed by the HEL uses two General Instruments AY-3-8912 programmable sound generator LSI circuits, a Digital Equipment Corporation (DEC) M1710 UNIBUS foundation module, and some control and signal conditioning circuitry. All the components are mounted on a DEC M1710 UNIBUS foundation module (see Figure 2) and connected to the UNIBUS on a DEC VAX 11/750 minicomputer.

THEORY OF OPERATION

Programmable Sound Generator (PSG)[1]

The AY-3-8912 is a register-oriented device consisting of 13 read and write registers (see Appendix B), each alterable under program control. The PSG can be subdivided into six basic blocks for producing sound (see Figure 3). They include:

a. Tone generators that produce square wave tones at user-controlled frequencies for each channel (A,B,C).

b. Noise generator that "produces a frequency-modulated pseudo random pulse width square wave for noise output."[2]

c. Mixer that combines the tone and noise for each output channel (A,B,C).

d. Amplitude controls that allow fixed or amplitude modulated volume control for each channel (A,B,C).

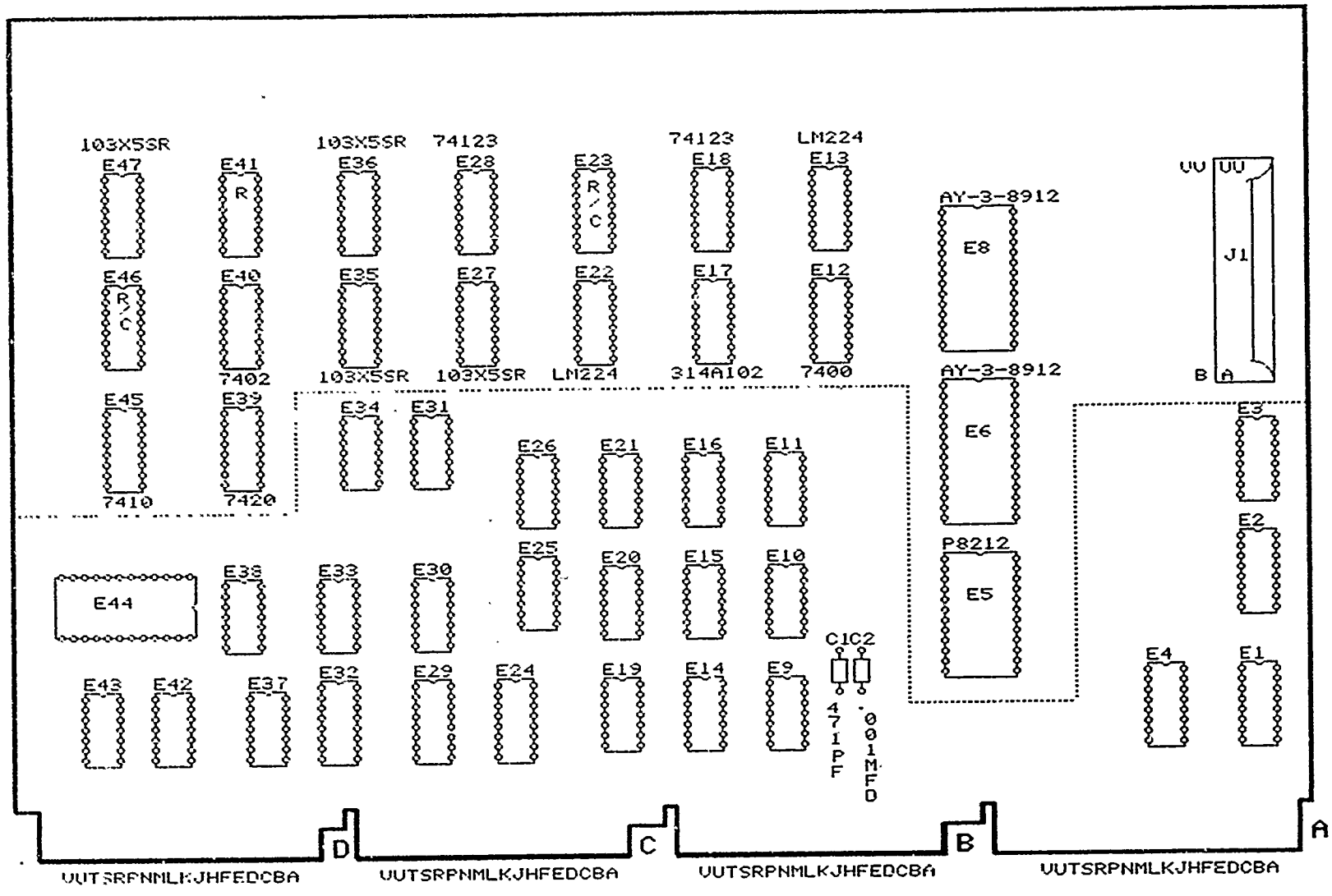
e. Envelope generator that selects one of 10 envelope patterns for shaping (modulating) the output.

f. Digital-to-Analog (D/A) converters that produce the sounds for each channel from the data loaded in the read and write registers on the PSG.

---

[1]The information about the PSG was developed from the detailed description and specifications pertaining to the AY-3-8912 Programmable Sound Generator. General Instruments Corporation, "Programmable Sound Generator," Microelectronics Data Catalog (1982) pp. 5.18 to 5.25.

[2]General Instruments, Microelectronics Data Catalog (1982) p. 5.19.



NOTES:
R - DISCRETE RESISTORS
R/C - DISCRETE RESISTOR/CAPCITORS

Figure 2. M1710 UNIBUS foundation module.

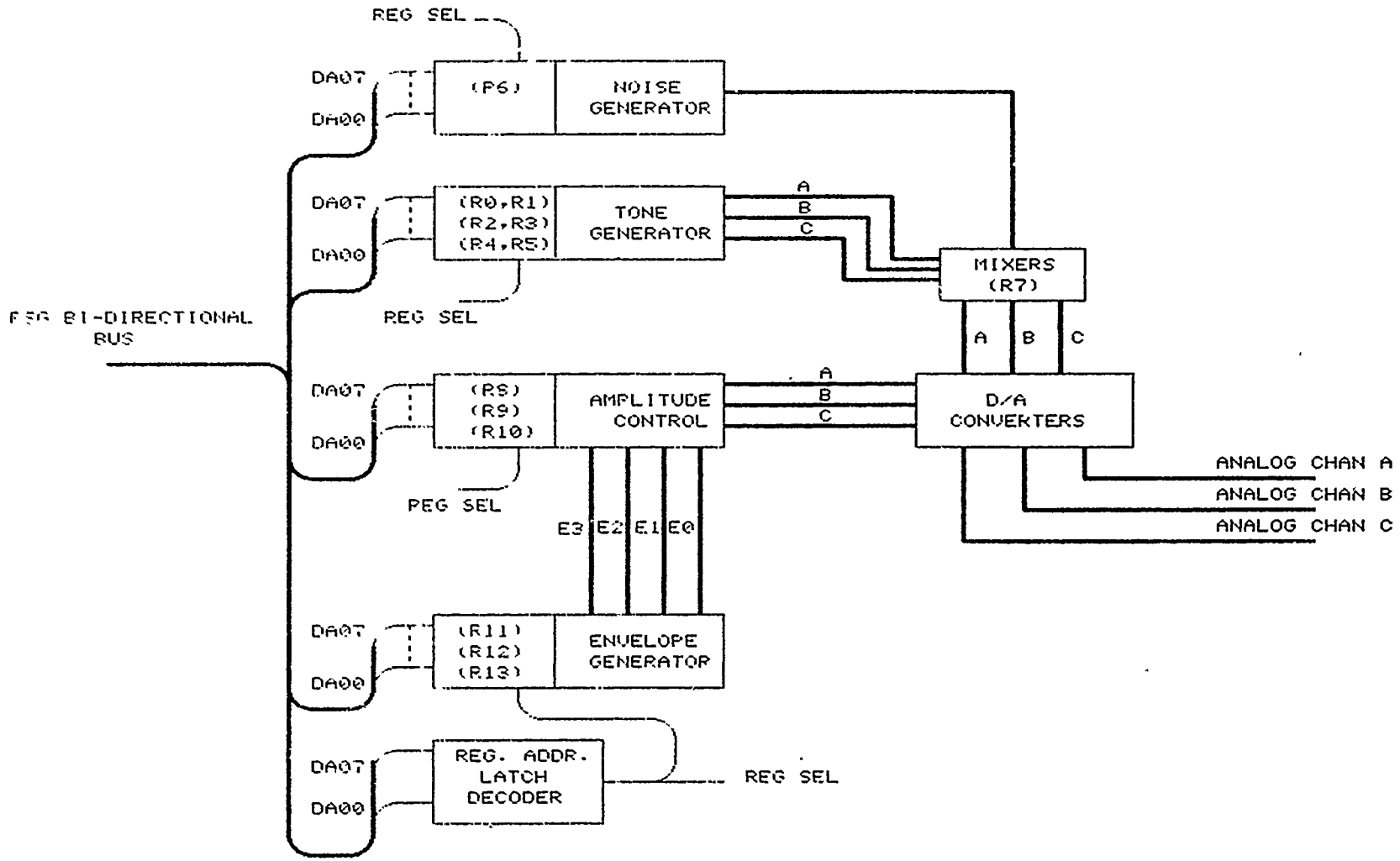


Figure 3. PSG block diagram.

Tone Control

There are six tone generator control registers, Register 0 (R0) through Register 5 (R5), that control the frequencies for the three output channels (A, B, and C) that produce tones. Each channel has two registers associated with it. The frequency for each channel is determined by the 8-bit count (0 to 255) loaded in the first register called the fine tune register and the 4-bit count (0 to 15) loaded in the second register called coarse tune register. Combined, the registers produce a 12-bit value used to divide the basic clock frequency of 87.5 KHz by the value contained in the corresponding registers for each channel (see Table 1). The 8-bit fine tune register contains the Least Significant Bits (LSB) and is used for making fine frequency adjustments of the tone. The four-bit coarse tune register contains the Most Significant Bits (MSB) and is used for making coarse frequency adjustments of the tone.

Table 1

Tone Register Channel Assignments

Channel	Coarse Tune Register	Fine Tune Register
A	R1	R0
B	R3	R2
C	R5	R4

Noise Generator Control

A single noise generator control register, Register 6 (R6), controls the frequency of the noise source. The noise frequency component is determined by the 5-bit value (range 0 to 31) loaded in R6 and is derived by dividing the basic noise clock frequency (87.5 KHz) by the value loaded in R6. Since there is only one noise generator control register, the noise frequency component on each channel will be the same for all three channels, but the amplitude of each may be individually controlled (see Amplitude Control section).

Mixer Control

A single mixer control register, Register 7 (R7), controls the tone and noise mixing for each of the three analog output channels on each PSG. The mixer can combine either, neither, or both tone and noise for each channel. The mixing is determined by the 6-bit value (range 0 to 63) loaded in R7. Bit 0 (B0), bit 1 (B1), and bit 2 (B2) designate which channels are enabled or disabled (channels A, B, and C) for tone, whereas, bit 3 (B3), bit 4 (B4), and bit 5 (B5) designate which channels are enabled or disabled for noise (see Table 2). To enable a channel requires that the appropriate bit(s) be reset or off.

Table 2

Mixer Control

Function	NOISE ENABLE	TONE ENABLE
Bit Number	B5 B4 B3	B2 B1 B0
Channel	C B A	C B A

Amplitude Control

Three 5-bit amplitude control registers, Register 8 (R8), Register 9 (R9), and Register 10 (R10), control the amplitude level and select the mode for each output channel (A, B, and C). Bit 4 in each amplitude control register designates whether the amplitude will be fixed (bit 4 = 0) or if the amplitude will be modulated (bit 4 = 1). When the mode is "fixed," bits 0 through 3 provide 16 (0 to 15) discrete volume levels with 0 being lowest level (off) and 15 being the highest. In "modulated" mode, the volume is controlled according to the shape and frequency produced by the envelope generator (see Envelope Generator Control section).

Envelope Generator Control

The envelope generator control section makes it possible to vary the envelope period or frequency as well as select or change the envelope shape when modulating the output channels (see Amplitude Control section).

Envelope Period Control

Two 8-bit envelope period control registers, Register 14 (R14) and register 15 (R15), control the period or frequency of the selected envelope pattern. R14, the envelope fine tune register, represents the LSB and provides the envelope period fine tuning adjustments. The envelope coarse tune register, R15, represents the MSB and provides the envelope period coarse tuning adjustments. Internally combined, the resulting 16-bit value is used to divide the fundamental noise clock frequency (5.47 KHz) and derive the envelope period or frequency.

Envelope Shape and Cycle Control

A single 4-bit envelope shape and cycle control register, Register 16 (R16), determines the envelope shape and cycle. Bits 0 through 3 are used to produce any one of 10 envelope shapes (see Appendix A). Each bit has a specific control function in the envelope generator (see Table 3) which causes one of the envelope shapes to be produced.

Table 3

Envelope Shape and Control Register

Bit	Function
0	HOLD
1	ALTERNATE
2	ATTACK
3	CONTINUE

Digital-to-Analog (D/A) Converters

The final step in generating sounds is transforming the digital data, as represented by the PSG registers, into the actual analog sound. This occurs in the D/A converter, where each PSG is capable of generating as many as three separate channels of sounds. Each channel is independently controlled and may contain a noise component or tone component as determined by the register data.

M1710 UNIBUS Foundation Module

The UNIBUS foundation module provides a general purpose, single-board interfacing module that supplies the basic interfacing circuitry when configuring custom devices to computers using the DEC UNIBUS architecture. The module provides the electronic circuitry for device address selection, interrupt generation, and data bus receiving and driving. Furthermore, the module can accommodate numerous 14-pin and 16-pin dual-in-line-packages (DIP) as well as some DIP to as many as 40 pins, providing the basic building block for custom interfaces.

Address Selection

The SEG responds to UNIBUS addresses 167724 through 167737 (octal). The execution of a data transfer instruction causes the appropriate address bits (A00 through A17) and the control bits (C0 and C01) to be placed on the UNIBUS (see timing diagram in Appendix C). Each device on the UNIBUS receives the address and control bits and begins the device address decoding. After a short interval delay for deskew, MSYN L is asserted by the master device or in this case the central processing unit (CPU). Each device on the UNIBUS receives MSYN L, completes its device address decoding, whereupon the device with the proper address responds by asserting DEVICE ENABLE L.

PSG Selection and Commands

DEVICE ENABLE L allows further decoding of address bits A01 through A04 by the 4- to 16-line decoder (see Figure 4). This causes the appropriate PSG to be selected and the proper PSG command to be initiated (see timing diagram in Appendix C): LATCH PSG, READ PSG, or WRITE PSG.

LATCH PSG selects one of the 13 registers associated with each PSG. SEL 30 (LATCH PSG1) selects the register specified by the data lines BUS D00 through BUS D15 (see Figure 5) for PSG1 by initiating the proper control signals BC2, BC1, BDIR, and SEL PSG1 (see Figure 4). SEL 36 (LATCH PSG2) initiates the same action for PSG2 (see Figure 4).

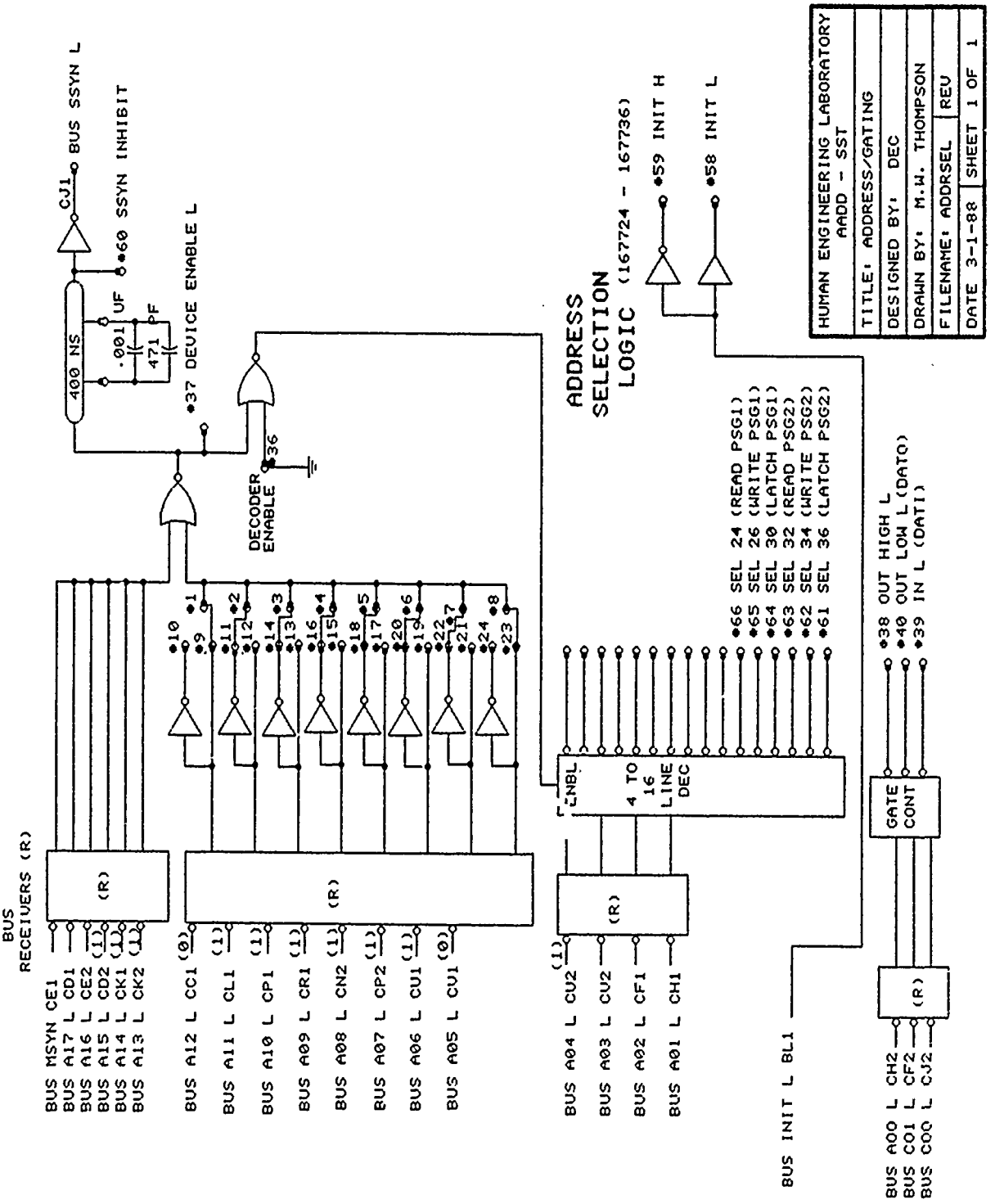
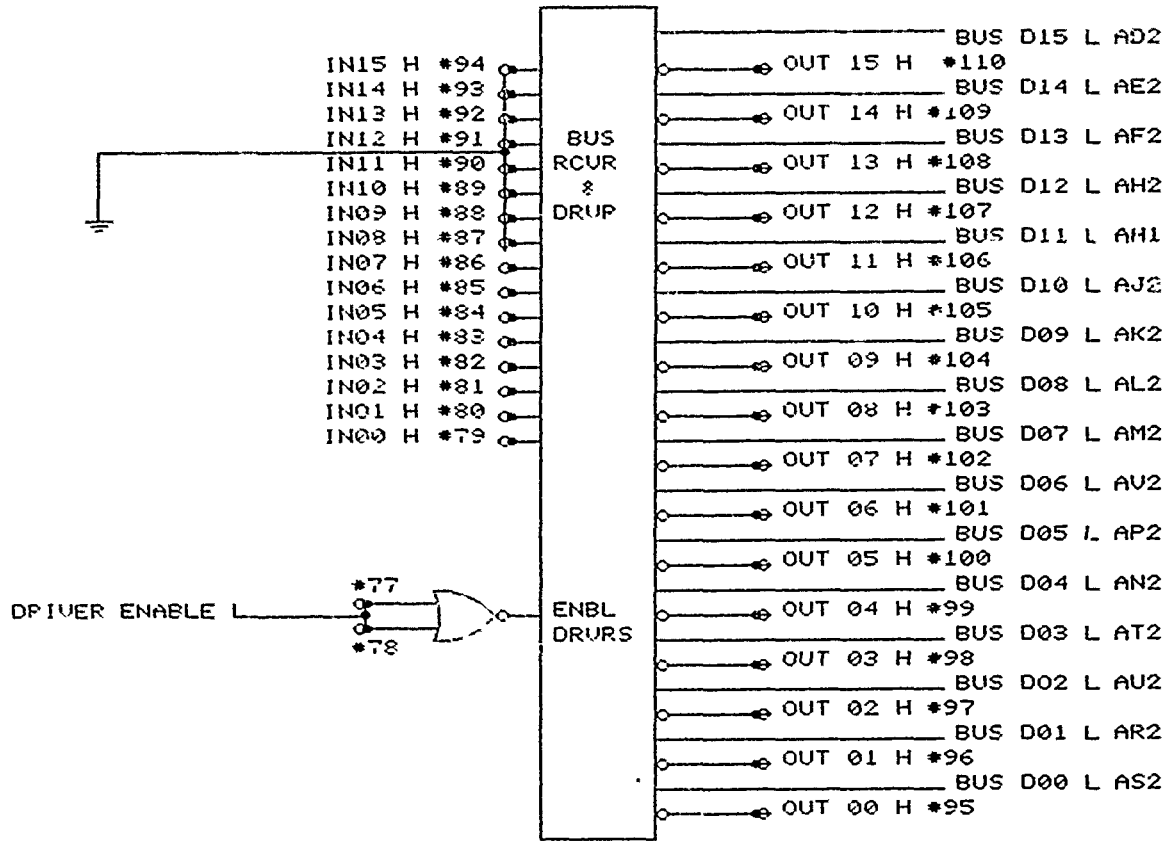


Figure 4. Address and gating control.



POWER

+5V AA2,BA2,CA2
GND ALL PINS C & T

HUMAN ENGINEERING LABORATORY AADD - SST	
TITLE: BUS DRIVER/RECEIVER	
DESIGNED BY: DEC	
DRAWN BY: M.W. THOMPSON	
FILENAME: RCR_DVR	REV
DATE 2/19/88	SHEET OF

Figure 5. Bus driver and receiver.

READ PSG allows the user to examine the contents of a PSG register previously selected by the LATCH PSG command. SEL 24 (READ PSG1) performs a register read of the register previously selected by initiating the proper control signals BC2, BC1, BDIR, and SEL PSG1 (see Figure 4) for PSG1. SEL 24 (READ PSG2) initiates the same action for PSG2.

WRITE PSG allows the user to deposit a value into a PSG register previously selected by the LATCH PSG command. SEL 26 (WRITE PSG1) deposits a value into the register previously selected by initiating the proper control signals BC2, BC1, BDIR, and SEL PSG1 (see Figure 4) for PSG2. SEL 34 (WRITE PSG2) initiates the same action for PSG2.

Gating Control

UNIBUS control lines C00 and C01 (see Figure 4) designate the type of data transaction, data-in (DATI) or data-out (DATO), with respect to the master device.

DATI transactions (C00 = 0 and C01 = 1) cause data transfers from the PSG register selected to the address specified by the processor. A DATI transaction occurs when a READ PSG command is executed by the CPU.

DATO transactions (C00 = 1 and C01 = 0) cause data transfers from an address specified by the processor to the PSG register selected. A DATO transaction occurs when a LATCH PSG or WRITE PSG command is executed by the CPU.

Data Bus Interface

The data bus interface contains the UNIBUS drivers, receivers, and 8-bit I/O PORT for establishing the two-way data path between the PSGs and the CPU. Data are transferred from the PSG to the CPU (READ PSG) by way of the PSG Bi-directional Bus (see Figure 1). Data for the CPU are placed on input lines IN00 through IN07 (see Figure 5), whereupon assertion of DRIVER ENBL L (see Figure 6) signal causes the data to be placed on the UNIBUS data lines for transfer to the CPU.

Data are transferred from the CPU to the PSG by way of UNIBUS data lines D15 through D00 and output lines OUT 15 through OUT 00 (see Figure 5). Data placed on data lines D15 through D00 are passed to the output line OUT 15 through OUT 00. These in turn are passed to the input lines of the 8-bit I/O PORT (see Figure 7) which places the data on the PSG Bi-directional Bus. The data are then available for processing by the selected PSG for either register selection (LATCH PSG) or data transfer to the selected register (WRITE PSG).

Amplifier and Mixer

The amplifier and mixer section of the SEG contains the signal conditioning, driving, and final mixing (adder) circuits for the SEG (see Figure 8). Each PSG output channel is connected to a signal driver whose output feeds an adder which combines the three output channels of each PSG. The output of the adder then feeds the signal conditioning circuit consisting of a voltage divider and AC coupling capacitor for final output.

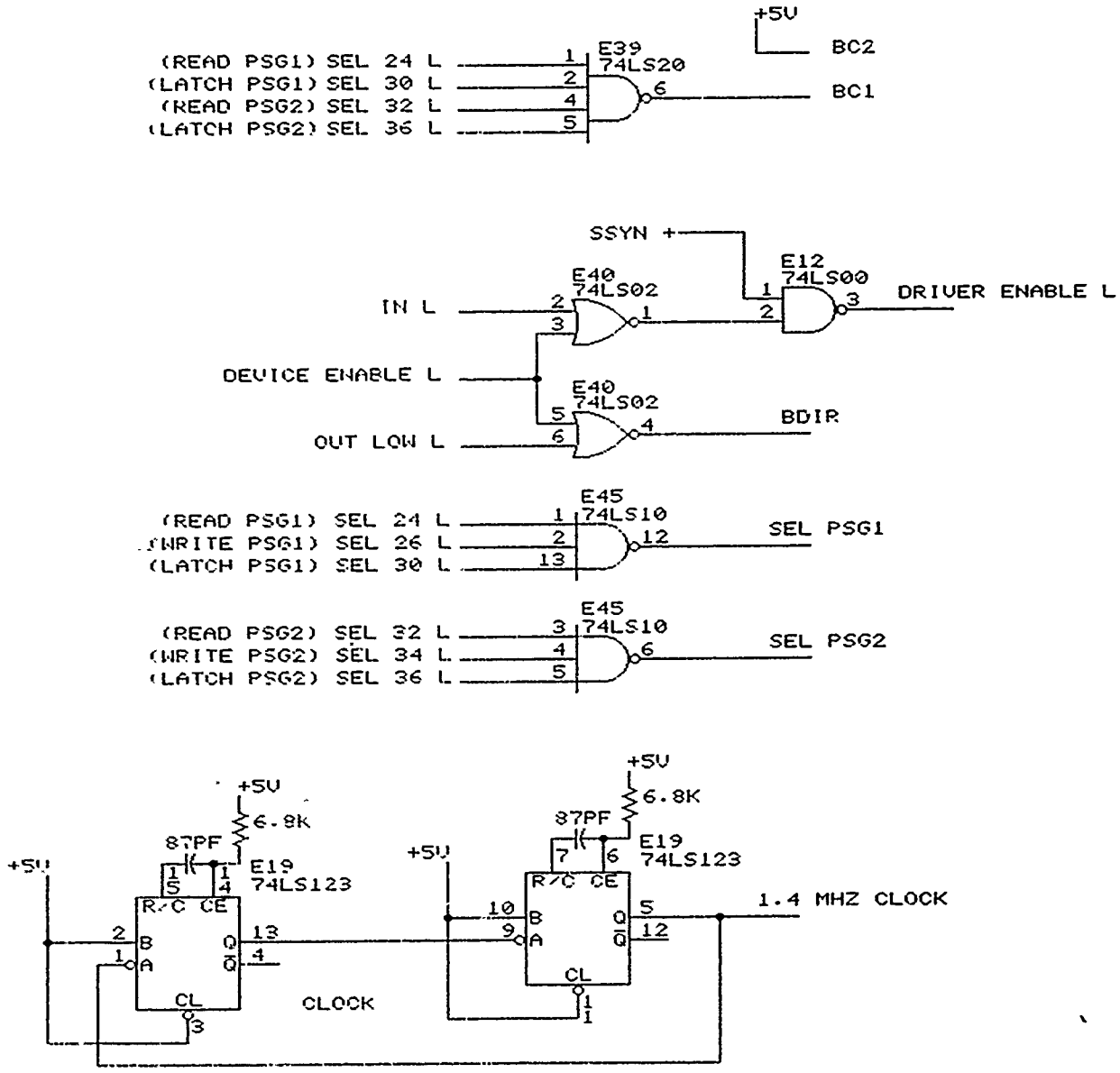


Figure 6. PSG control logic.

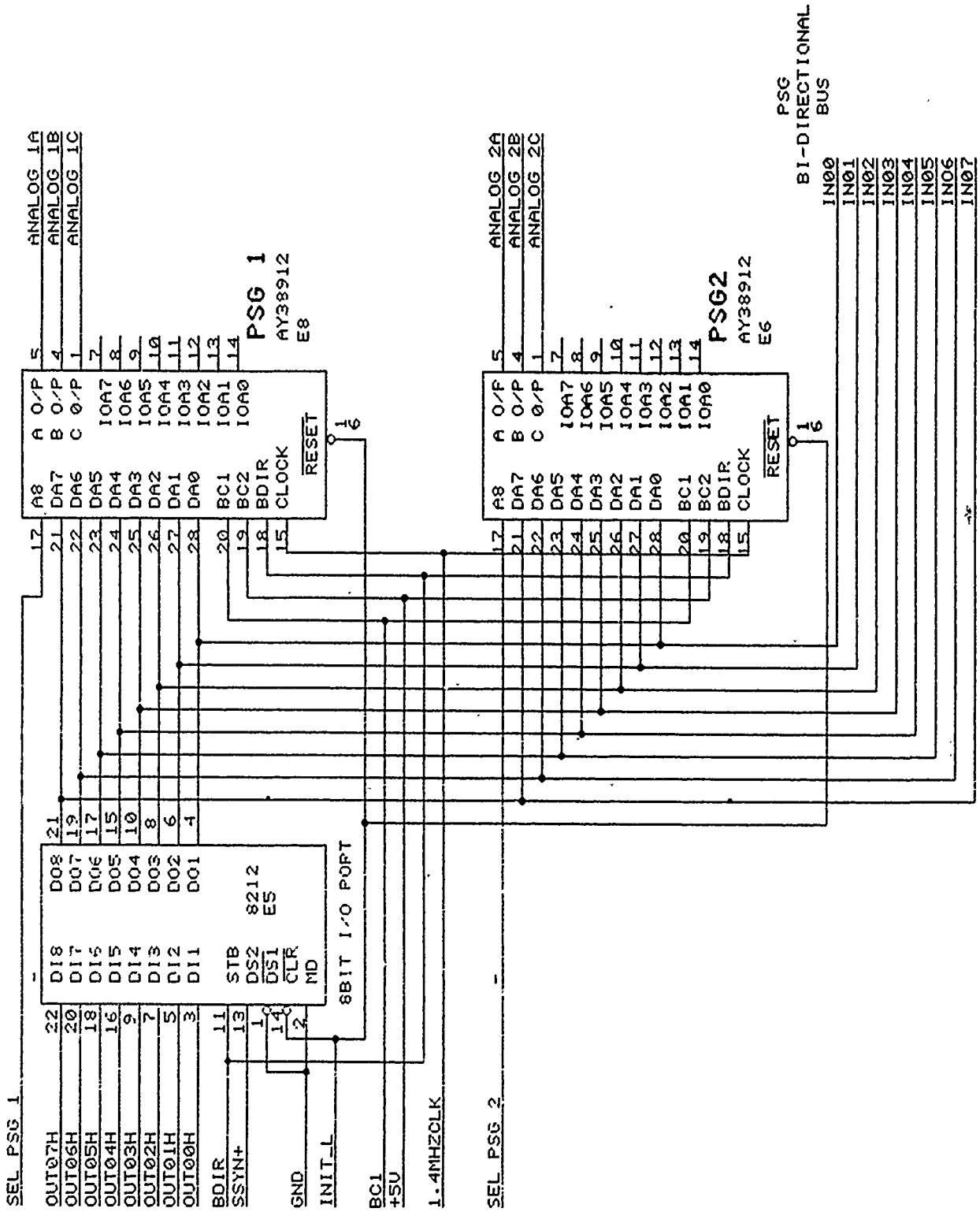


Figure 7. PSG bi-directional bus.

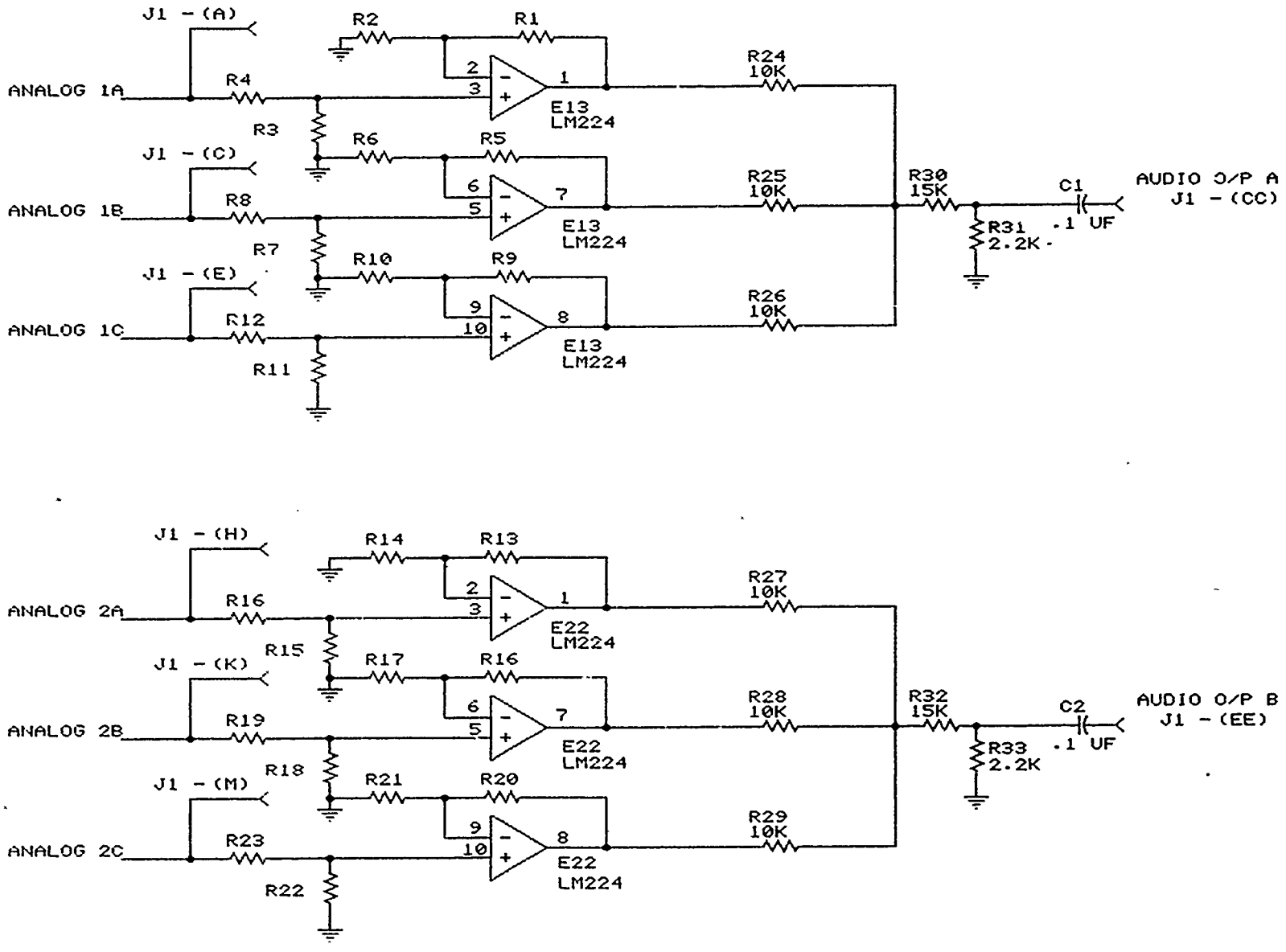


Figure 8. PSG amplifier and mixer.

SOUND EFFECTS GENERATOR USER'S GUIDE

Software has been developed for use with the sound effects generator (SEG). The software provides easy access to the SEG with inclusion of FORTRAN callable MACRO subroutines (see Appendix D). The Macro source listings for each subroutine are provided in Appendix E and a working FORTRAN example program in Appendix F. Using the subroutines requires adhering to the proper calling format in your application program. Then you must link those subroutines with your application program. The SEG object files required for linking are on the VAX 11/750 in the directory DRA0:[SEG.OBJ].

Two important concepts should be noted pertaining to the SEG. First, the sounds produced by the programmable sound generators (PSG) on the SEG are initiated and changed by simply performing a series of register loads. Therefore, each functional area used in producing sounds (tone, noise, mixing, amplitude, and envelope generation) can be related to a specific register within each PSG.

Second, once a sound has been generated by the PSG in the modulated mode, that sound is sustained without further intervention by the controlling process. Since the PSG is a register-oriented device, it only responds to changes made in its internal registers. This feature reduces the overhead required by any process when interacting with the SEG since generating or changing a particular sound requires setting only a few data registers while repetitive sounds require no intervention.

All the subroutines used to access the SEG require mapping to a global page frame section. This section should have been created before running any software using the PSG software. The procedure for installing the page frame section is described in the following "Create and Map" section. One final note: all numeric values should be declared as integers.

Create and Map Section

For the subroutines to work, a page frame section must be created for the SEG so that users are able to map to the section created for the PSG. In essence, the process creates a group global section called PSGSEC, which allows processes to associate (map) a section of its address space with the specific physical device addresses of the SEG. The page frame section can be installed by running the program:

```
$RUN PSGMAP
```

The process requires the following privileges:

PFNMAP to create a page frame section.

PRMGBL to create a permanent global section.

Register Reset

A subroutine has been developed for FORTRAN inclusion that will reset or clear all the registers (0 through 13) for either programmable sound generator (PSG) or both on the SEG. The calling format is:

```
CALL PSG_RESET (PSG_NUM)
```

In which the PSG_NUM argument has the following meaning:

PSG_NUM = 0 reset both PSG1 and PSG2.
PSG_NUM = 1 reset PSG1.
PSG_NUM = 2 reset PSG2.

Tone Selection

Each PSG on the SEG has three separate channels that may be programmed with a different tone. Associated with each channel are two registers for determining the tone frequency. The first register represents the LSB, bits 0 through 7, and is used for fine tuning the tone frequency. The second register represents the MSB, bits 8 through 11, and is used for coarse tuning the tone frequency. Combined, they produce a 12-bit number, which is used to divide the fundamental tone clock frequency (87.5 KHz) down to the tonal frequency required.

The subroutine developed for FORTRAN inclusion has the following format:

CALL PSG_TONE (PSG_NUM, CHAN_NUM, COARSE_ADJ, FINE_ADJ)

The arguments have the following significance:

PSG_NUM	specifies which PSG will be modified, 1 or 2.
CHAN_NUM	specifies which channel (1, 2, or 3) for setting the tonal frequency.
COARSE_ADJ	is the value of bits 8 through 11 for the coarse frequency adjustments. The range is 0 to 15.
FINE_ADJ	is the value of bits 0 through 7 for the fine frequency adjustments. The range is 0 to 255.

Noise Selection

Each PSG on the SEG has three channels that may be programmed with noise. The frequency content of the noise for all channels is determined by this one register, the noise generator control register. Again, as with the tone generation, the frequency content is determined by dividing the fundamental clock frequency (87.5 KHz) by the value of the 5-bit noise generation control register.

The subroutine developed for FORTRAN inclusion has the following format:

CALL PSG_NOISE (PSG_NUM, NOISE_FREQ)

The arguments have the following significance:

PSG_NUM	specifies which PSG will be modified and has a value of 1 or 2.
NOISE_FREQ	is the value representing bits 0 through 4 used for calculating the noise frequency content and has a value ranging from 0 to 31.

Mixer Selection

Each PSG on the SEG has a mixer that will blend the tone and noise for each channel on the PSG. Blending or mixing is controlled by the mixer control register, an 8-bit register, of which the lower 6 bits are used for mixing the tone and noise. This is accomplished by constructing a bit mask in the mixer control register, in which bit 0 through bit 2 form the tone mixer mask, and bit 3 through bit 5 form the noise mixer mask for channels 1, 2, and 3. Mixing is disabled when the respective bit for tone or noise is set or enabled when the respective bit for tone or noise is reset. For example, if bit 0 is set, there is no tone on channel 1, or if bit 3 is set, there is no noise on channel 1. The same convention is followed for channels 2 and 3.

The subroutine developed for FORTRAN inclusion has the following format:

CALL PSG_MIXER (PSG_NUM, MIXER_MASK)

The arguments have the following significance:

PSG_NUM	specifies which PSG will be modified and has a value of 1 or 2.
MIXER_MASK	is the value representing the mixer mask, bits 0 through 5, enabling or disabling tone and noise mixing for each channel. The value range is 0 through 63.

Amplitude Control

Each PSG on the SEG has three channels that may be programmed independently for amplitude (volume). The amplitude of each channel is determined by the 5-bit value in their respective amplitude control register. Each amplitude control register has two modes of operation associated with it: a fixed mode and a modulated mode. The mode is determined by bit 5, the "mode" select bit. In the fixed mode, bit 5 ("mode" select) is reset, and bit 0 through bit 4 determine one of 16 discrete steps for volume control where 0 is off and 15 is maximum. In the modulated mode, bit 5 ("mode" select) is set, and bit 0 through bit 5 are ignored. The volume is then modulated as determined by the envelope generator described in the next section "Envelope Generator Control."

The subroutine developed for FORTRAN inclusion has the following format:

CALL PSG_AMP (PSG_NUM, CHAN_NUM, AMPLITUDE)

The arguments have the following significance:

PSG_NUM	specifies which PSG will be modified, and has the value of 1 or 2.
CHAN_NUM	specifies the channel (1, 2, or 3) for volume setting or modulation selection.
AMPLITUDE	is the value representing bits 0 through bit 5 for setting the volume level. The value is 16 for modulation or ranges between 0 and 15 for fixed volumes.

Envelope Generator Control

When in modulated mode (see "Amplitude Control" section), "mode" select bit is set, and each PSG output on the SEG can be controlled by the envelope generator. The envelope generator allows the selection of a particular wave form pattern (see Appendix A) and allows for setting the period of the wave form. The pattern is used to create the envelope shape for the desired output.

Envelope Period Control

The envelope period control determines the frequency or period of the envelope pattern. Associated with the envelope period are two registers for determining the envelope period. The first register represents the LSB, bit 0 through bit 7, and is called the envelope fine tune register. The second register represents the MSB, bits 8 through bit 15, and is called the envelope coarse tune register. Combined, they produce a 16-bit value, which is used to divide the envelope clock frequency of 5.47 KHz down to the envelope period desired.

The subroutine developed for FORTRAN inclusion has the following format:

CALL PSG_ENV_GEN (PSG_NUM, COARSE_VAL, FINE_VAL)

The arguments have the following significance:

PSG_NUM	specifies which PSG will be modified and has a value of 1 or 2.
COARSE_VAL	is the value representing bit 8 through bit 15 in the envelope coarse tune adjustment register. The value range is from 0 to 255.
FINE_VAL	is the value representing bit 0 through bit 7 in the envelope fine tune adjustment register. The value range is from 0 to 255.

Envelope Shape and Cycle Control

The envelope shape and cycle control determines the envelope shape and defines the envelope as a single or repetitive (cycling) event. A 4-bit register, called the envelope shape and cycle control register, selects one of 10 different wave forms (see Appendix A). The wave form selected is used to create the output shape (envelope), which becomes a single burst or repetitive bursts of noise or tones.

The subroutine developed for FORTRAN inclusion has the following format:

CALL PSG_ENV_SHAPE (PSG_NUM, SHAPE_VAL)

The arguments have the following significance:

PSG_NUM	specifies which PSG will be modified and has a value of 1 or 2.
SHAPE_VAL	is the value representing bit 0 through bit 3 for the envelope shape and cycle control register. The value range is from 0 to 15 (see Appendix A for patterns).

Reading PSG Registers

It is possible to examine the contents of any PSG register. A subroutine has been developed for FORTRAN inclusion that will read the contents of the specified register. The calling format is

CALL PSG_READ (PSG_NUM, REG_NUM, REG_VAL)

In which the arguments have the following significance:

PSG_NUM	specifies which PSG will be modified and has a value of 1 or 2.
REG_NUM	is the number of the register to be read (0 to 13). The range is from 0 to 13 (see Appendix B for the number of the register functions).
REG_VAL	is the value of the specified register.

REFERENCES

Digital Equipment Corporation. (1978). Computer interfacing accessories and logic handbook. Maynard, MA: Digital Equipment Corporation.

Digital Equipment Corporation. (1979). PDP11bus handbook. Maynard, MA: Digital Equipment Corporation.

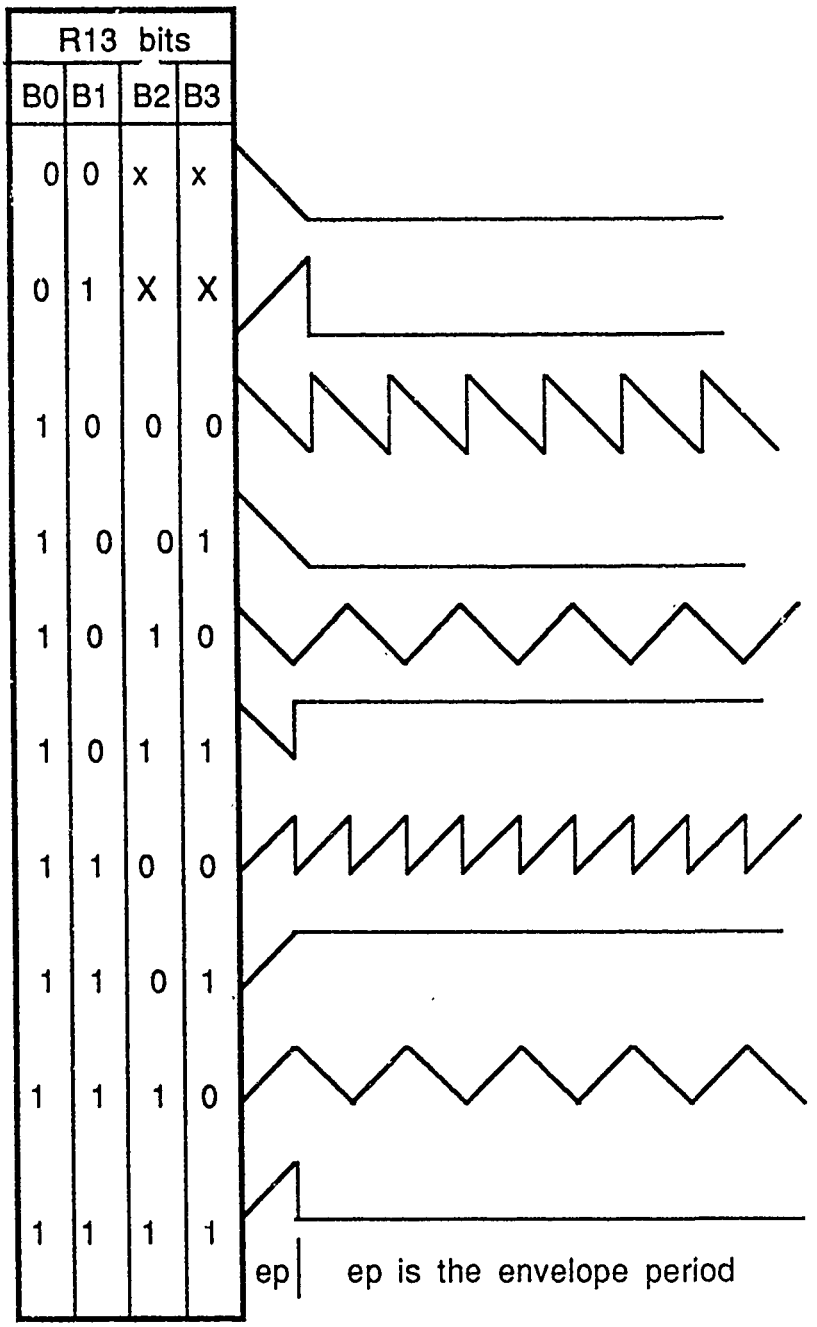
Digital Equipment Corporation. (1982). VAX hardware handbook. Maynard, MA: Digital Equipment Corporation.

General Instrument Corporation. (1982). Microelectronics data catalog. Hicksville, NY: General Instrument Corporation.

APPENDIX A

ENVELOPE SHAPE AND CYCLE PATTERNS

ENVELOPE SHAPE AND CYCLE PATTERNS[3]



3 General Instruments Corporation. Programmable Sound Generator," Microelectronics Data Catalog (Hi ville, N.Y. 1982) p. 5.22.

APPENDIX B

PROGRAMMABLE SOUND GENERATOR REGISTER FUNCTIONS

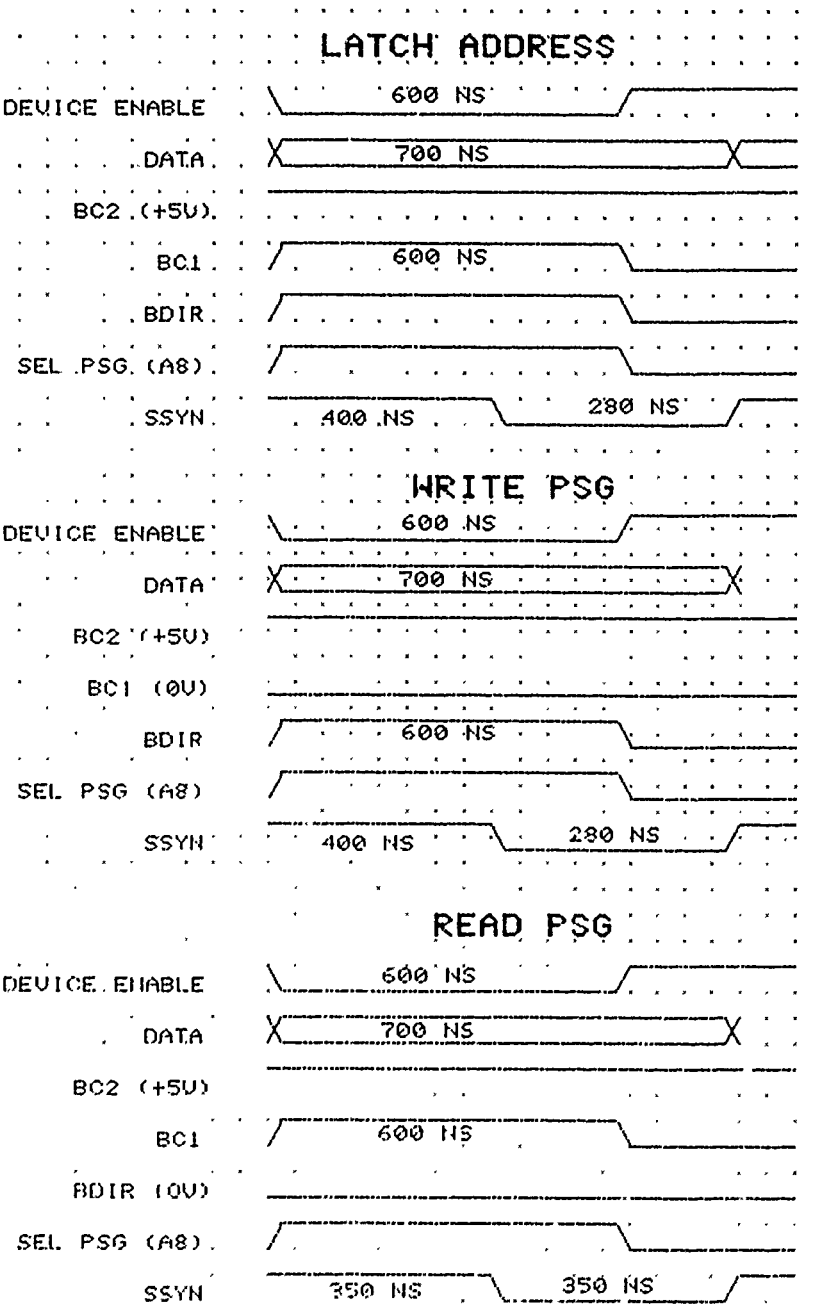
PROGRAMABLE SOUND GENERATOR REGISTER FUNCTIONS

Register	Function
R0	Channel A Tone Period (fine)
R1	Channel A Tone Period (coarse)
R2	Channel B Tone Period (fine)
R3	Channel B Tone Period (coarse)
R4	Channel C Tone Period (fine)
R5	Channel C Tone Period (coarse)
R6	Noise Period
R7	Mixer
R8	Channel A Amplitude Control
R9	Channel B Amplitude Control
R10	Channel C Amplitude Control
R11	Envelope Period (fine)
R12	Envelope Period (coarse)
R13	Envelope Shape and Cycle Control

APPENDIX C

PSG TIMING DIAGRAM

PSG TIMING DIAGRAM



APPENDIX D

SOUND EFFECTS GENERATOR PROGRAMS AND SUBROUTINES

SOUND EFFECTS GENERATOR PROGRAMS AND SUBROUTINES

FORTRAN PROGRAMS

PSGMAP - creates the global page frame section for the Sound Effects Generator.

FORTRAN SUBROUTINES

Subroutine	Arguments
PSG_RESET	(psg_num)
PSG_TONE	(psg_ num, chan_num, coarse_adj, fine_adj)
PSG_NOISE	(psg_num, noise_freq)
PSG_MIXER	(psg_num, mixer_mask)
PSG_AMP	(psg_num, chan_num, amplitude)
PSG_ENV_GEN	(psg_num, coarse_val, fine_val)
PSG_ENV_SHAPE	(psg_num, shape)
PSG_READ	(psg_num, reg_num, reg_value)

APPENDIX E

MACRO SOURCE CODE FOR FORTRAN CALLABLE SUBROUTINES

DRA1:[THOMPSON.PSG]PSGMAP.MAR;64

```
;*****************************************************************************
; This is a FORTRAN callable macro subroutine that returns the
; Creates the map section PAGE FRAME MAP (PFN).
; Programmable Sound Generator (PSG) UNIBUS address:
;       register            offset      address
;       --------            ------      -------
;       READ PSG1           ^O0         ^O767724    ^X1F7EA3
;       WRITE PSG1          ^O2         ^O767726
;       LATCH PSG1          ^O4         ^O767730

;       READ PSG1           ^O6         ^O767732
;       WRITE PSG1          ^O10        ^O767734
;       LATCH PSG1          ^O12        ^O767736

;*****************************************************************************
; $CRMPSC -- Create and Map Section:
;*****************************************************************************
; see 'SYSTEM SERVICES, I/0' in the memory management services section.
; FORMAT (MACRO): $CRMPSC [inadr],[retadr],[acmode],[flags],
;                         [gsdnam],[ident],[relpag],[chan],
;                         [pagcnt],[vbn],[prot],[pfc]
;*****************************************************************************
;   Arguments required for creating the page frame section:

;       inadr - Starting and ending virtual addresses into which the
;               section is to be mapped.  If the starting and ending
;               addresses are the same, a single page is mapped.

;       retadr- Starting and ending virtual addresses into which the section
;               was actually mapped.

;       flags - Flag mask specifying the type of section to be mapped or
;               created. (#SEC$M_WRT!SEC$M_PFNMAP)

;       pagcnt- Number of pages in the section. Cannot equal zero for physical
;               page frame.

;       vbn   - Specifies the page frame number where the section begins in
;               memory.  ((base addr. + unibus offset) / bytes per page) where

;               base addr. = unibus base addr. UB0 = ^X 20100000
;               unibus offset = force stick begins at ^O 767724
;               bytes per page = 512
;*****************************************************************************
;       USE BYTE ADDRESSING MODE:

;               UBA_BASE  = @RETRANGE  = ^X 2013EFD4

;               READ  PSG1 = @RETRANGE  = ^X 2013EFD4
;               WRITE PSG1 = @RETRANGE+2 = ^X 2013EFD6
;               LATCH PSG1 = @RETRANGE+4 = ^X 2013EFD8
```

```
;               READ  PSG2 = @RETRANGE+6 = ^X 2013EFDA
;               WRITE PSG2 = @RETRANGE+8 = ^X 2013EFDC
;               LATCH PSG2 = @RETRANGE+10= ^X 2013EFDD

; The following assign should also work for VBN if a literal (#) is
implied:
;       VBN1 = <^X20100000 + ^O767724> @-9

; Create and map page frame section.
;*****************************************************************************

        .TITLE PSG_MAP

        .PSECT MAP_SECTION ,NOEXE,WRT,RD,PIC

MAPRANGE:
        .LONG       1024
        .LONG       1024

RETRANGE:
        .BLKL       2                   ; reserve two longwords

REMAINDER:
        .LONG       468
LATCH_REG:
        .LONG       0
REG_VAL:
        .LONG       0


        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF

        .PSECT      CODE ,EXE,NOWRT

        .ENTRY      PSGMAP ^M<>

        VBN1 = <IO750$AL_UB0SP+^O767724>@-9

        $CRMPSC_S -
                INADR = MAPRANGE,-
                RETADR = RETRANGE,-
                FLAGS = #SEC$M_WRT!SEC$M_PFNMAP!SEC$M_EXPREG,-
                PAGCNT = #1,-
                VBN = #<IO750$AL_UB0SP+^O767724>@-9

        CLRL R5
        CLRL R6

START:
        ADDL3       REMAINDER,RETRANGE,R3; get first address
```

```
PSG1:
        MOVAB       4[R3],R4        ; latch ADDR
        MOVAB       2[R3],R5        ; write ADDR
        MOVL    #11,R2

LATCH:
        MOVB    R2,(R4)             ; select register
        MOVB    R2,LATCH_REG
;       BRW     LATCH
WRITE:
        MOVB    R2,(R5)     ; set selected reg.
;       ADDB2   #1,R2
;       BRW     WRITE
READ:
        MOVB        (R3),REG_VAL        ; Get data from latched register
        SUBB3   #15,R2,R1
;       BRW     READ

        BLEQ    LATCH
;       BRW     PSG1

DONE:
        $EXIT_S
        .END PSGMAP

DRA1:[THOMPSON.PSG]PSG_AMP.MAR;7

;*****************************************************************************
; AUTHOR: MICHAEL W. THOMPSON
; DATE  : 17 JUNE 1987
;*****************************************************************************
; DESCRIPTION:      Programmable Sound Generator (PSG) amplitude control.
;                   registers 10,11,12 for channels A,B,C, respectively.

; This is a FORTRAN callable macro subroutine that:

; Calling format: CALL PSG_AMP (PSG_NUM,CHAN_NUM,AMPLITUDE)

;       Selects the PSG1 or PSG2      : PSG_NUM       (1 or 2)
;       Selects the channel           : CHAN_NUM (1,2,3)
;       Set the amplitude for channel
;       selected.                     : AMPLITUDE

; PSG UNIBUS address:
;       register            offset      address
;       --------            ------      -------
;       READ PSG1           ^O0         ^O767724    ^X1F7EA3
;       WRITE PSG1          ^O2         ^O767726
;       LATCH PSG1          ^O4         ^O767730

;       READ PSG1           ^O6         ^O767732
;       WRITE PSG1          ^O10        ^O767734
;       LATCH PSG1          ^O12        ^O767736
```

```
;*****************************************************************************
        .TITLE P_S_G_AMP
;*****************************************************************************

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF

        .PSECT MAP_SECTION,NOEXE,WRT,RD,PIC
MAPRANGE:
        .LONG       1024
        .LONG       1024
RETRANGE:
        .BLKL       2                   ; reserve two longwords

        .PSECT DATA,NOEXE,WRT,RD,PIC
REMAINDER:
        .LONG       468
PSG_NUM:
        .LONG       1
CHAN_NUM:
        .LONG       0
AMPLITUDE:
        .LONG       3
NAME:
        .ASCID      /PSGSEC/

        .PSECT      CODE,EXE,NOWRT
        .ENTRY      PSG_AMP, ^M<R3,R4,R5,IV>

;MAP to permanent global page frame section (I/O page)
        $MGBLSC_S -
              INADR  = MAPRANGE,-
              RETADR = RETRANGE,-
              FLAGS  = #SEC$M_WRT!SEC$M_EXPREG,-
              GSDNAM = NAME
BEGIN:
        ADDL3       REMAINDER,RETRANGE,R3 ; put PSG1 base address in R3

        MOVL        @4(AP),PSG_NUM      ; get PSG number
        MOVL        @8(AP),CHAN_NUM   ; get CHANNEL number
        MOVL        @12(AP),AMPLITUDE ; get CHANNEL number

        CMPB        #2,PSG_NUM          ; sel PSG1 or PSG2
        BEQL PSG2

PSG1:
        MOVAB       4[R3],R4      ; put latch ADDR in R4
        MOVAB       2[R3],R5      ; put write ADDR in R5
        BRW  CALC_CHAN

PSG2:
        MOVAB       10[R3],R4     ; put latch ADDR in R4
        MOVAB       8[R3],R5      ; put write ADDR in R5
```

```
CALC_CHAN:
        CMPB        #1,CHAN_NUM
        BEQL  CHAN_A
        CMPB        #2,CHAN_NUM
        BEQL  CHAN_B
CHAN_C:
        MOVB        #^O12,CHAN_NUM
        BRW   SET_AMP
CHAN_B:
        MOVB        #^O11,CHAN_NUM
        BRW   SET_AMP
CHAN_A:
        MOVB        #^O10,CHAN_NUM

SET_AMP:
        MOVB        CHAN_NUM,(R4)   ; latch AMPLITUDE register
        MOVB        AMPLITUDE,(R5)  ; set amplitude register
DONE:
        RET
        .END

;       $EXIT_S
;       .END PSG_AMP
```

DRA1:[THOMPSON.PSG]PSG_ENV_GEN.MAR;8

```
;*****************************************************************************
; AUTHOR: MICHAEL W. THOMPSON
; DATE  : 17 JUNE 1987
;*****************************************************************************
; DESCRIPTION:  Programmable Sound Generator (PSG) envelope generator
;               control register 13 (fine tune) & register 14 (coarse tune).

; This is a FORTRAN callable macro subroutine that:

; Calling format:
;       CALL PSG_ENV_GEN (PSG_NUM,COURSE_VAL,FINE_VAL)

;       Selects the PSG1 or PSG2        : PSG_NUM       (1 or 2)
;       Sets the course adjust tune value: COURSE       (^O 0 THRU ^O 377)
;       Sets the fine adjust tune value  : FINE         (^O 0 thru ^O 377)

; PSG UNIBUS address:
;       register        offset      address
;       --------        ------      -------
;       READ PSG1       ^O0         ^O767724    ^X1F7EA3
;       WRITE PSG1      ^O2         ^O767726
;       LATCH PSG1      ^O4         ^O767730

;       READ PSG1       ^O6         ^O767732
;       WRITE PSG1      ^O10        ^O767734
;       LATCH PSG1      ^O12        ^O767736
```

```
;*****************************************************************************
        .TITLE P_S_G_ENV_GEN
;*****************************************************************************

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF

        .PSECT MAP_SECTION,NOEXE,WRT,RD,PIC
MAPRANGE:
        .LONG       1024
        .LONG       1024
RETRANGE:
        .BLKL       2                   ; reserve two longwords

        .PSECT DATA,NOEXE,WRT,RD,PIC
REMAINDER:
        .LONG       468
LATCH_REG:
        .LONG       0
REG_VAL:
        .LONG       0
PSG_NUM:
        .LONG       1
COARSE:
        .LONG       3
FINE:
        .LONG       7
NAME:
        .ASCID      /PSGSEC/

        .PSECT      CODE,EXE,NOWRT
        .ENTRY      PSG_ENV_GEN, ^M<R3,R4,R5,IV>

        $MGBLSC_S -
                INADR = MAPRANGE,-
                RETADR = RETRANGE,-
                FLAGS = #SEC$M_WRT!SEC$M_EXPREG,-
                GSDNAM = NAME

BEGIN:
        ADDL3       REMAINDER,RETRANGE,R3 ;put PSG1 base address in R3

        MOVL        @4(AP),PSG_NUM          ; get PSG number
        MOVL        @8(AP),COARSE   ; get COARSE value
        MOVL        @12(AP),FINE    ; get FINE value

        CMPB        #2,PSG_NUM      ; sel PSG1 or PSG2
        BEQL        PSG2

PSG1:
        MOVAB       4[R3],R4        ; latch ADDR
        MOVAB       2[R3],R5        ; write ADDR
        BRW  LATCH
```

```
PSG2:
        MOVAB       10[R3],R4     ; latch ADDR
        MOVAB       8[R3],R5      ; write ADDR

LATCH:
        MOVB        #^O13,(R4)
        MOVB        FINE,(R5)
        MOVB        #^O14,(R4)
        MOVB        COARSE,(R5)
DONE:
;       BRW  LATCH
        RET
        .END

;       $EXIT_S
;       .END PSG_ENV_GEN

DRA1:[THOMPSON.PSG]PSG_ENV_SHAPE.MAR;8

;*****************************************************************************
; AUTHOR: MICHAEL W. THOMPSON
; DATE  : 17 JUNE 1987
;*****************************************************************************
; DESCRIPTION:     Programmable Sound Generator (PSG) envelope shape and cycle
;                  control (register 15).

; This is a FORTRAN callable macro subroutine that:

; Calling format: CALL PSG_ENV_SHAPE (PSG_NUM,SHAPE_VAL)

;       Selects the PSG1 or PSG2      : PSG_NUM       (1 or 2)
;       Selects the shape control     : SHAPE_VAL     (1 thru ^O 17)

;        Shape control bit functions:
;                B0 - Hold
;                B1 - Alternate
;                B2 - Attack
;                B3 - Continue

; PSG UNIBUS address:
;       register            offset      address
;       --------            ------      -------
;       READ PSG1           ^O0         ^O767724    ^X1F7EA3
;       WRITE PSG1          ^O2         ^O767726
;       LATCH PSG1          ^O4         ^O767730

;       READ PSG1           ^O6         ^O767732
;       WRITE PSG1          ^O10        ^O767734
;       LATCH PSG1          ^O12        ^O767736
```

```
;*****************************************************************************
        .TITLE P_S_G_ENV_SHAPE
;*****************************************************************************

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF

        .PSECT MAP_SECTION,NOEXE,WRT,RD,PIC
MAPRANGE:
        .LONG       1024
        .LONG       1024
RETRANGE:
        .BLKL       2                   ; reserve two longwords

        .PSECT DATA,NOEXE,WRT,RD,PIC
REMAINDER:
        .LONG       468
PSG_NUM:
        .LONG       1
SHAPE_VAL:
        .LONG       0
NAME:
        .ASCID      /PSGSEC/

;***** BEGIN PSG_NOISE *****
        .PSECT      CODE,EXE,NOWRT
        .ENTRY      PSG_ENV_SHAPE, ^M<R3,R4,R5,IV>

        $MGBLSC_S -
              INADR = MAPRANGE,-
              RETADR = RETRANGE,-
              FLAGS = #SEC$M_WRT!SEC$M_EXPREG,-
              GSDNAM = NAME

BEGIN:
        ADDL3       REMAINDER,RETRANGE,R3 ; put PSG1 base address in R3

        MOVL        @4(AP),PSG_NUM          ; get PSG number
        MOVL        @8(AP),SHAPE_VAL  ; get shape control value

        CMPB        #2,PSG_NUM        ; sel PSG1 or PSG2
        BEQL        PSG2

PSG1:
        MOVAB       4[R3],R4      ; put latch ADDR in R4
        MOVAB       2[R3],R5      ; put write ADDR in R5
        BRW  LATCH

PSG2:
        MOVAB       10[R3],R4     ; put latch ADDR in R4
        MOVAB       8[R3],R5      ; put write ADDR in R5
```

```
LATCH:
        MOVB        #^O15,(R4)   ; select register 15 (envelope shape control)
        MOVB        SHAPE_VAL,(R5)

DONE:

;       BRW  LATCH
        RET
        .END

;       $EXIT_S
;       .END PSG_ENV_SHAPE
```

DRA1:[THOMPSON.PSG]PSG_MAP.MAR;12

```
;*****************************************************************************

;                               PSG_MAP.MAR

; This is a FORTRAN callable macro subroutine
; Creates a global map section PAGE FRAME MAP (PFN).
; This allows users to map to the physical addresses associated
; with the programmable sound generators.
;
; Calling format:  CALL PSG_MAP
;
; Programmable Sound Generator (PSG) UNIBUS address:
;       register            offset      address
;       --------            ------      -------
;       READ PSG1           ^O0         ^O767724    ^X1F7EA3
;       WRITE PSG1          ^O2         ^O767726
;       LATCH PSG1          ^O4         ^O767730

;       READ PSG1           ^O6         ^O767732
;       WRITE PSG1          ^O10        ^O767734
;       LATCH PSG1          ^O12        ^O767736

;*****************************************************************************
; $CRMPSC -- Create and Map Section system service
;*****************************************************************************
; see 'SYSTEM SERVICES, I/0' in the memory management services section.
;
; FORMAT (MACRO):  $CRMPCS [inadr],[retadr],[acmode],[flags],
;                          [gsdnam],[ident],[relpag],[chan],
;                          [pagcnt],[vbn],[prot],[pfc]
;*****************************************************************************
;   Arguments required for creating the page frame section:

;       inadr - Starting and ending virtual addresses into which the
;               section is to be mapped.  If the starting and ending
;               addresses are the same, a single page is mapped.
```

```
;       retadr- Starting and ending virtual addresses into which the section
;               was actually mapped.

;       flags - Flag mask specifying the type of section to be mapped or
;               created. (#SEC$M_WRT!SEC$M_PFNMAP)

;       pagcnt- Number of pages in the section. Cannot equal zero for physical
;               page frame.

;       vbn   - Specifies the page frame number where the section begins in
;               memory.  ((base addr. + unibus offset) / bytes per page) where

;               base addr. = unibus base addr. UB0 = ^X 20100000
;               unibus offset = force stick begins at ^O 767724
;               bytes per page = 512
;*****************************************************************************
;       USE BYTE ADDRESSING MODE:

;               UBA_BASE  = @RETRANGE  = ^X 2013EFD4

;               READ  PSG1 = @RETRANGE  = ^X 2013EFD4
;               WRITE PSG1 = @RETRANGE+2 = ^X 2013EFD6
;               LATCH PSG1 = @RETRANGE+4 = ^X 2013EFD8

;               READ  PSG2 = @RETRANGE+6 = ^X 2013EFDA
;               WRITE PSG2 = @RETRANGE+8 = ^X 2013EFDC
;               LATCH PSG2 = @RETRANGE+10= ^X 2013EFDD

; The following assign should also work for VBN if a literal (#) is
implied:
;       VBN1 = <^X20100000 + ^O767724> @-9
;*****************************************************************************

        .TITLE PSG_MAP

        .PSECT MAP_SECTION ,NOEXE,WRT,RD,PIC

MAPRANGE:
        .LONG       1024
        .LONG       1024

RETRANGE:
        .BLKL       2                   ; reserve two longwords

REMAINDER:
        .LONG       468
LATCH_REG:
        .LONG       0
REG_VAL:
        .LONG       0

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF
```

```
        .PSECT      CODE ,EXE,NOWRT

        .ENTRY      PSGMAP ^M<>

        VBN1 = <IO750$AL_UB0SP+^O767724>@-9

; Create and map page frame section.
        $CRMPSC_S -
                INADR = MAPRANGE,-
                RETADR = RETRANGE,-
                FLAGS = #SEC$M_WRT!SEC$M_PFNMAP!SEC$M_EXPREG,-
                PAGCNT = #1,-
                VBN = #<IO750$AL_UB0SP+^O767724>@-9

        $EXIT_S
        .END PSGMAP
```

DRA1:[THOMPSON.PSG]PSG_MIXER.MAR;6

```
;*****************************************************************************
; AUTHOR: MICHAEL W. THOMPSON
; DATE  : 17 JUNE 1987
;*****************************************************************************
; DESCRIPTION:  Programmable Sound Generator (PSG) mixer control I/O
; enable. (register 7)

; This is a FORTRAN callable macro subroutine that:

; Calling format: CALL PSG_MIXER (PSG_NUM,MIXER_MASK)

;       Selects the PSG1 or PSG2        : PSG_NUM       (1 or 2)
;       Selects the noise and tone channels
;       to be mixed                     : MIXER_MASK    (^O1 thru ^O177)

; PSG UNIBUS address:
;       register            offset      address
;       --------            ------      -------
;       READ PSG1           ^O0         ^O767724    '·X1F7EA3
;       WRITE PSG1          ^O2         ^O767726
;       LATCH PSG1          ^O4         ^O767730

;       READ PSG1           ^O6         ^O767732
;       WRITE PSG1          ^O10        ^O767734
;       LATCH PSG1          ^O12        ^O767736

;*****************************************************************************
        .TITLE P_S_G_MIXER
;*****************************************************************************

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF
```

```
        .PSECT MAP_SECTION,NOEXE,WRT,RD,PIC
MAPRANGE:
        .LONG       1024
        .LONG       1024
RETRANGE:
        .BLKL       2                   ; reserve two longwords

        .PSECT DATA,NOEXE,WRT,RD,PIC
REMAINDER:
        .LONG       468
PSG_NUM:
        .LONG       1
MIXER_MASK:
        .LONG       0
NAME:
        .ASCID      /PSGSEC/

        .PSECT      CODE,EXE,NOWRT
        .ENTRY      PSG_MIXER, ^M<R3,R4,R5,IV>

        $MGBLSC_S -
              INADR  = MAPRANGE,-
              RETADR = RETRANGE,-
              FLAGS  = #SEC$M_WRT!SEC$M_EXPREG,-
              GSDNAM = NAME

BEGIN:
        ADDL3       REMAINDER,RETRANGE,R3 ; put PSG1 base address in R3

        MOVL        @4(AP),PSG_NUM        ; get PSG number
        MOVL        @8(AP),MIXER_MASK ; mixer mask

        CMPB        #2,PSG_NUM        ; sel PSG1 or PSG2
        BEQL        PSG2

PSG1:
        MOVAB       4[R3],R4      ; put latch ADDR in R4
        MOVAB       2[R3],R5      ; put write ADDR in R5
        BRW  LATCH
PSG2:
        MOVAB       10[R3],R4     ; put latch ADDR in R4
        MOVAB       8[R3],R5      ; put write ADDR in R5
LATCH:
        MOVB        #7,(R4)               ; select register 7
        MOVB        MIXER_MASK,(R5)
DONE:
;       BRW  LATCH
        RET
        .END

;       $EXIT_S
;       .END PSG_MIXER
DRA1:[THOMPSON.PSG]PSG_NOISE.MAR;7
```

```
;*****************************************************************************
; AUTHOR: MICHAEL W. THOMPSON
; DATE  : 17 JUNE 1987
;*****************************************************************************
; DESCRIPTION:     Programmable Sound Generator (PSG)

; This is a FORTRAN callable macro subroutine that:

; Calling format: CALL PSG_NOISE (PSG_NUM,NOISE_FREQ)

;       Selects the PSG1 or PSG2      : PSG_NUM       (1 or 2)
;       Selects the NOISE_FREQ            : CHAN_NUM       (1,2,3 for A,B,C)

; PSG UNIBUS address:
;       register            offset          address
;       --------            ------          -------
;       READ PSG1           ^O0             ^O767724    ^X1F7EA3
;       WRITE PSG1          ^O2             ^O767726
;       LATCH PSG1          ^O4             ^O767730

;       READ PSG1           ^O6             ^O767732
;       WRITE PSG1          ^O10            ^O767734
;       LATCH PSG1          ^O12            ^O767736

;*****************************************************************************

        .TITLE P_S_G_NOISE

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF

        .PSECT MAP_SECTION,NOEXE,WRT,RD,PIC
MAPRANGE:
        .LONG       1024
        .LONG       1024
RETRANGE:
        .BLKL       2                   ; reserve two longwords

        .PSECT DATA,NOEXE,WRT,RD,PIC
REMAINDER:
        .LONG       468
PSG_NUM:
        .LONG       1
NOISE_FREQ:
        .LONG       6
NAME:
        .ASCID      /PSGSEC/

;***** BEGIN PSG_NOISE *****
        .PSECT      CODE,EXE,NOWRT
        .ENTRY      PSG_NOISE, ^M<R3,R4,R5,IV>
```

```
        $MGBLSC_S -
                INADR = MAPRANGE,-
                RETADR = RETRANGE,-
                FLAGS = #SEC$M_WRT!SEC$M_EXPREG,-
                GSDNAM = NAME

BEGIN:
        ADDL3       REMAINDER,RETRANGE,R3 ;put PSG1 base address in R3

        MOVL        @4(AP),PSG_NUM          ; get PSG number
        MOVL        @8(AP),NOISE_FREQ ; get CHANNEL number

        CMPB        #2,PSG_NUM          ; sel PSG1 or PSG2
        BEQL        PSG2

PSG1:
        MOVAB       4[R3],R4      ; put latch ADDR in R4
        MOVAB       2[R3],R5      ; put write ADDR in R5
        BRW  LATCH

PSG2:
        MOVAB       10[R3],R4     , put latch ADDR in R4
        MOVAB       8[R3],R5      ; put write ADDR in R5

LATCH:
        MOVB        #6,(R4)               ; select register 6 (noise gen. control)
        MOVB        NOISE_FREQ,(R5)

DONE:

;       BRW  LATCH
        RET
        .END

;       $EXIT_S
;       .END PSG_NOISE

DRA1:[THOMPSON.PSG]PSG_READ.MAR;22

;*****************************************************************************
; AUTHOR: MICHAEL W. THOMPSON
; DATE  : 17 JUNE 1987
;*****************************************************************************
; DESCRIPTION:     Programmable Sound Generator (PSG) read from selected
register.

; This is a FORTRAN callable macro subroutine that:

; Calling format:
;       CALL PSG_READ (PSG_NUM,REG_NUM,REG_VALUE)

;       Selects the PSG1 or PSG2       : PSG_NUM       (1 or 2)
;       Selects the register number    : REG_NUM       (1 thru 15)
;       Value returned from register   : REG_VALUE     (0 THRU 256)
```

```
; PSG UNIBUS address:
;       register            offset      address
;       --------            ------      -------
;       READ PSG1           ^O0         ^O767724    ^X1F7EA3
;       WRITE PSG1          ^O2         ^O767726
;       LATCH PSG1          ^O4         ^O767730

;       READ PSG1           ^O6         ^O767732
;       WRITE PSG1          ^O10        ^O767734
;       LATCH PSG1          ^O12        ^O767736

;*****************************************************************************

        .TITLE P_S_G_READ

;*****************************************************************************

        .LIBRARY /SYS$LIBRARY:LIB.MLB/
        $IO750DEF

        .PSECT MAP_SECTION,NOEXE,WRT,RD,PIC
MAPRANGE:
        .LONG       1024
        .LONG       1024
RETRANGE:
        .BLKL       2                   ; reserve two longwords

        .PSECT DATA,NOEXE,WRT,RD,PIC
REMAINDER:
        .LONG       468
PSG_NUM:
        .LONG       1
REG_NUM:
        .LONG       0
REG_VAL:
        .LONG       0
NAME:
        .ASCID      /PSGSEC/

        .PSECT      CODE,EXE,NOWRT
        .ENTRY      PSG_READ, ^M<R3,R4,R5,IV>

        $MGBLSC_S -
              INADR = MAPRANGE,-
              RETADR = RETRAN
```

APPENDIX F

FORTRAN PROGRAM EXAMPLE

```
c*************************************************************************

c       PROCESS NAME:  ENG_SOUND.FOR
c       AUTHOR:  Michael W. Thompson
c       DATE:  29 Feb. 1988

c*************************************************************************
DESCRIPTION:  This program is an example of using the SEG and is provided as a working
example of generating a particular sound.  This program produces a sound somewhat similar to
the engine sounds produced by helicopters.  The sound produced contains a high pitched whine,
mixed with pulsating noise.  Adjustments of tone and noise were done by ear.  The program
examines the torque value, conditions the value for SEG, then changes the frequency component
of the noise simulating pitch blade changes that occur when changes are made with the collective
control.

c*************************************************************************

c  Subroutines used have the following calling conventions:

c       CALL PSG_RESET(PSG_NUM)
c       CALL PSG_TONE (PSG_NUM,CHAN_NUM,COARSE_ADJ,FINE_ADJ)
c       CALL PSG_NOISE (PSG_NUM,NOISE_FREQ)
c       CALL PSG_MIXER (PSG_NUM,MIXER_MASK)
c       CALL PSG_AMP (PSG_NUM,CHAN_NUM,AMPLITUDE)
c       CALL PSG_ENV_GEN (PSG_NUM,CHAN_NUM,COARSE_TUNE,FINE_TUNE)
c       CALL PSG_ENV_SHAPE (PSG_NUM,SHAPE_VAL)
c       CALL PSG_READ (PSG_NUM,REG_NUM,REG_VAL)

c  The arguments have the following significance:

c               PSG_NUM   = PSG number (1 or 2).
c               CHAN_NUM = tone channel number (1,2,OR 3).
c               COARSE_ADJ = coarse tone adjust value.
c               FINE_ADJ = fine tone adjust value.
c               NOISE_FREQ = noise frequency.
c               MIXER_MASK = bit representation for tone or noise disable = 1.
c               AMPLITUDE = 16 for env. gen. control; 0 - 15 for fixed amp.
c               COARSE_TUNE = envelope period coarse tune adjust.
c               FINE_TUNE = envelope period fine tune adjust.
c               SHAPE_VAL = envelope waveform pattern select.
c               REG_NUM = register number (0 - 13)
c               REG_VAL = returned register value.

c*************************************************************************

        PROGRAM ENG_SOUND

        IMPLICIT INTEGER (A-Z)

        REAL*4  T1,T2,T3
        REAL*4  PERCENT_TORQUE,OLD_TORQUE
        REAL*4  TORQUE
```

```
      INTEGER*4 PSG_NUM,CHAN_NUM
      INTEGER*4 COARSE,FINE,NOISE_FREQ
      INTEGER*4 AMPLITUDE,SHAPE_VAL,COARSE_TUNE
      INTEGER*4 FINE_TUNE,MIXER_MASK
      INTEGER*4 STATUS
      INTEGER*4 BINARY_INTERVAL(2)

      LOGICAL  DONE

      CHARACTER ascii_interval*10/'0 0:0:0.30'/

C... BEGIN Engine sound

C... Initialize clock
      T1 = SECNDS(0.0)

c... open input data file
      open (unit = 1,file = 'eng_sound.dat',status = 'OLD')

c... read the data
      read (1,1001)PSG_NUM,CHAN_NUM,
     &          COARSE,FINE,NOISE_FREQ,
     &          AMPLITUDE,SHAPE_VAL,COARSE_TUNE,
     &          FINE_TUNE,MIXER_MASK

1001  format (I10)

201   CONTINUE

c Use the print statement for debugging and variable examination; otherwise omit comment.

c     print *, PSG_NUM,CHAN_NUM,
c    &          COARSE,FINE,NOISE_FREQ,
c    &          AMPLITUDE,SHAPE_VAL,COARSE_TUNE,
c    &          FINE_TUNE,MIXER_MASK

C Call macro programs to reset registers then get X & Y values

200   continue
      CALL PSG_MIXER (PSG_NUM,MIXER_MASK)

c... Set up rotor noise sound
      chan_num = 1
      amplitude = 16
      CALL PSG_NOISE (PSG_NUM,NOISE_FREQ)
      CALL PSG_ENV_GEN (PSG_NUM,COARSE_TUNE,FINE_TUNE)
      CALL PSG_ENV_SHAPE (PSG_NUM,SHAPE_VAL)
      CALL PSG_AMP (PSG_NUM,CHAN_NUM,AMPLITUDE)

c... Set up tone sound
      chan_num = 2
      amplitude = 4
      CALL PSG_TONE (PSG_NUM,CHAN_NUM,COARSE,FINE)
      CALL PSG_AMP (PSG_NUM,CHAN_NUM,AMPLITUDE)
```

```
c... Set up background noises
        chan_num = 3
        amplitude = 7
        CALL PSG_AMP (PSG_NUM,CHAN_NUM,AMPLITUDE)

        I = 0

c...  Check the PSG registers.
        print *,' '
        do while (I .LT. 14)
           CALL PSG_READ (1,I,REG_VAL1)
           CALL PSG_READ (2,I,REG_VAL2)
            PRINT *, 'REG ',I,' = ',REG_VAL1,REG_VAL2
           I = I + 1
        end do

c.. Convert ascii time to binary time
        status = sys$bintim (ascii_interval, binary_interval)
        if (.not. status) call lib$stop(%val(status))

c.. Schedule a wakeup
        status = sys$schdwk (,,binary_interval, binary_interval)
        if (.not. status) call lib$stop(%val(status))

c.. What for completion
        do while (.not. done)
c.. Let's hibernate
          status = sys$hiber()
          if (.not. status) call lib$stop(%val(status))

c... Check the value of torque
        print *,'noise freq = ',noise_freq
        print *,'Enter torque as a decimal value less than 1 = '
        read (5,1002)torque
1002    format (f4.2)

c... Check to see if the torque has changed.
          if (old_torque .NE. torque) then

            percent_torque = torque * 100.
            if (torque .LT. .33 ) then
             noise_freq = percent_torque / 8.
             CALL PSG_NOISE (PSG_NUM,NOISE_FREQ)

            else if (torque .LT. .66) then
                  noise_freq = percent_torque / 8.
                  CALL PSG_NOISE (PSG_NUM,NOISE_FREQ)

                else ! torque .LT. .99
                   noise_freq = percent_torque / 8.
                   CALL PSG_NOISE (PSG_NUM,NOISE_FREQ)
                 end if
             end if
```

```
 old_torque = torque

end do

CALL EXIT
STOP
END
```